Carnegie Mellon
Software Engineering Institute

Safety-Critical Systems and the TSP

Watts S. Humphrey

November 2005

Software Engineering Process Management

Unlimited distribution subject to the copyright.

Technical Note
CMU/SEI-2005-TN-011

Safety-Critical Systems and the TSP

Watts S. Humphrey

November 2005

Software Engineering Process Management

Technical Note
CMU/SEI-2005-TN-011

This work is sponsored by the U.S. Department of Defense.

The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Contents

Abstract v

1 Introduction 1

2 Background and Methods 3
- 2.1 The STAMP Method 3
- 2.2 Process Development and STAMP 5
- 2.3 Praxis High Integrity Systems 5

3 Implications for the TSP 6

4 Conclusion 7

Bibliography 9

List of Figures

Figure 1: STAMP Model of a Safety-Critical System 4

Abstract

Because the Team Software Process (TSP) has proven effective for developing high-quality software applications, a brief review of the safety-critical systems field has been conducted to determine whether the TSP could be usefully extended into this area. This technical note provides a brief overview of recent work in software safety, discusses the problems and implications of using the TSP for developing safety-critical systems, and presents some conclusions. This information is relevant to software developers and acquirers of safety-critical software-intensive systems.

1 Introduction

The Team Software Process[SM] (TSP[SM]) has proven effective for developing high-quality software applications. As a result, a brief review of the safety-critical systems field was conducted to determine whether the TSP could be usefully extended into this area. This technical note provides a brief overview of recent work in software safety, discusses the problems and implications of using TSP for developing such systems, and presents the following conclusions:

1. While defective software increases risk in safety-critical systems, the principal software safety problems found to date existed in systems that performed according to their designs. Unfortunately, the designs themselves were defective so that properly functioning software resulted in unsafe system operation.
2. The principal work in software safety has concerned accident analysis using Failure Modes and Effects Analysis (FMEA) methods. These so-called "event chain models" are effective for assigning liability for accidents and are widely used to support post-accident litigation. However, these methods are not as useful for preventing accidents through improved system analysis and design.
3. Until recently, little work had been done on applying and codifying safety-critical methods for designing and enacting software development processes.
4. More recent work on the Systems Theoretic Accident Modeling and Process (STAMP) method has provided the foundational ideas needed for designing software processes for safety-critical system development.
5. The use of STAMP-based methods by the TSP could usefully enhance the TSP process and contribute to broadening the use of the STAMP method itself.
6. While the TSP as presently defined could and should be used for developing safety-critical systems, if the TSP process were augmented to conform to STAMP principles, product safety and development performance should be substantially enhanced.
7. While safety accidents have only occasionally resulted from flawed software, the increasing use of software-intensive systems in continuously-operating safety-critical systems increases the need for improved software development practices. The work of Praxis High Integrity Systems is particularly suitable for such systems. Praxis is currently exploring the use of their methods in conjunction with the TSP. Should they decide to use the TSP, collaboration with the Software Engineering Institute (SEI) would facilitate and augment their efforts.

If suitable support could be obtained, the SEI should expand the focus of the TSP initiative to apply STAMP principles to developing, enacting, and managing systems development processes. The SEI could also work cooperatively with Praxis High Integrity Systems to jointly apply and enhance their combined methods.

2 Background and Methods

With the growing use of software in safety-critical systems, there has been a corresponding growth in accidents that, at least in part, have been blamed on the systems' software. Because these accidents often injure people and damage businesses, there has been an increasing volume of litigation. Not surprisingly, this has resulted in a growing need to identify accident causes, assign liability for the events, and recover damage costs. This in turn has led to methods for investigating accidents and much of this work has been based on the domino theory.

The domino theory postulates that each unsafe or out-of-control event has one or more causes and that these causes could also be unsafe or out-of-control events, which similarly have causes. By tracing these chains of causes and events backward, the investigator could presumably find an original cause of the out-of-control events and thus establish liability for the entire chain of events. A large volume of work has been based on these principles, including many accident reports and technical papers [Benediktsson 01, Goddard 00, Madan 97, Maier 97, Stalhane 98].

Some of the better-known safety-critical analysis methods are Hazard and Operability Studies (HAZOP), Fault Tree Analysis (FTA), Event Tree Analysis (ETA), Failure Modes and Effects Analysis (FMEA), Fault Hazard Analysis (FHA), Subsystem Hazard Analysis (SSHA), State Machine Hazard Analysis (SMHA), and Software Fault Tree Analysis (SFTA) [Leveson 95]. While these methods are generally useful for analyzing safety events after they happen, the recently developed STAMP method appears to be most suitable for guiding early safety-critical system design and development.

2.1 The STAMP Method

Because liability-focused investigations tend to stop as soon as a plausible culprit is identified, and because most complex events typically have many contributing causes, event chain-based methods do no generally identify the full set of conditions responsible for accidents. For example, a pilot might make an incorrect entry in a navigational instrument, causing an airliner to crash into a mountain. While the initial investigation assigns blame to the pilot, it could be that the charts the pilot used were confusing or out of date, or that the navigational beacons were confusingly named, or that the beacons were placed in unsafe locations. Because safety incidents involving complex systems generally have complex causes, the event chain accident model often prematurely stops the event-cause search when the first likely culprit is found.

The STAMP method, developed by the Massachusetts Institute of Technology's (MIT) Nancy Leveson, recognized the need for a more comprehensive view [Leveson 95, Leveson 02a, Leveson 02b, Leveson 02c]. In the STAMP method, the entire event is viewed as a feedback-control system, as shown in Figure 1. This feedback control model helps the investigator identify the many factors that contribute to unsafe conditions and to define the control actions needed to keep all system parameters within safe limits. Since the STAMP method is not designed to assign liability (and in fact is likely to exculpate the initially selected culprit), it is not likely to be generally adopted to support litigation. However, it is valuable for identifying systemic issues and preventing their recurrence.

A further aspect of the STAMP method that is particularly important is the recognition that systems evolve. When a system is initially developed, delivered, and operated, it may be completely safe. However, as the system is enhanced, as system operational practices evolve, and as the system environment changes, the system's operating parameters gradually migrate toward unsafe values. Leveson points out that unless such systems are periodically assessed and upgraded, all such systems ultimately end up operating at the very edge of criticality, and any of a host of possible events could trigger a safety incident [Leveson 02c]. While an event chain investigation might then identify an immediate cause of the specific event, that specific cause was merely a coincidence, and preventing its recurrence would not likely make the system measurably safer.

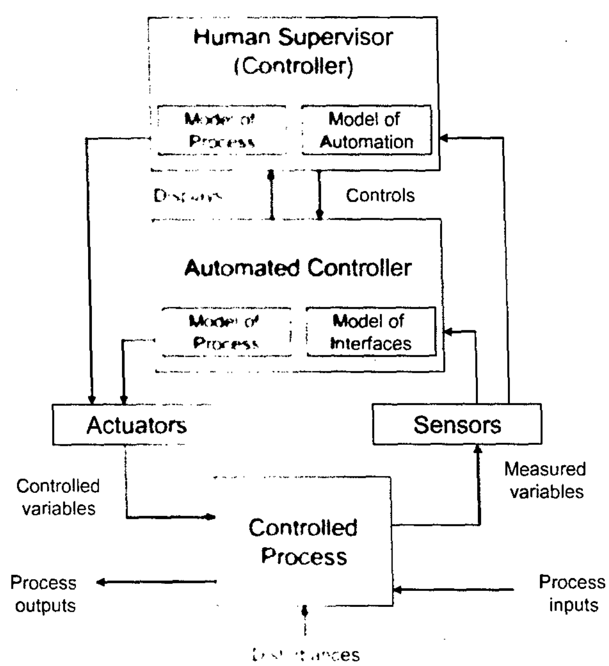

Figure 1: STAMP Model of a Safety-Critical System

2.2 Process Development and STAMP

The STAMP method has initially been used by Leveson and others to illuminate accident investigations [Johnson 03, Leveson 02a, Leveson 02b, Leveson 02c]. However, while the method has been of demonstrable value for post-accident investigations, as of yet there are no published examples of its use during system development. Since the method is well defined and appears to be readily extendable to development, this shortcoming should soon be corrected.

The most interesting aspect of the STAMP method from a process perspective is that it represents safety-critical systems as feedback-control models, and defines a theoretical framework and set of principles for designing risk management processes to control them. Further, since the STAMP method defines safety incidents as events that result in undesirable costs (and software and systems development processes can be viewed as processes to manage risk and reduce costs), the STAMP method could be viewed as a generalized method for analyzing and designing processes, and in particular, for designing development processes.

2.3 Praxis High Integrity Systems

Based on a review of the safety-critical design methods developed by Praxis High Integrity Systems located in Bath, England, it is clear that current common software development practices are inadequate for designing and developing high-integrity, software-intensive systems. The principal reasons are as follows.

1. Continuously operating safety-critical systems cannot be shut down, even when errors or failures occur.
2. Methods have been developed for using redundant equipment to protect safety-critical systems against hardware failures. These methods have generally been effective.
3. Use of redundant software systems has not been found effective, even when the redundant systems are independently developed. The principal reason is that common design errors tend to be repeated in the multiple redundant designs.
4. The Praxis methods apply rigorous analysis with special software support tools to produce software that is provably correct. These methods have been shown to produce exceedingly high-quality software that is comparable in quality to that produced by experienced TSP teams. Generally, the Praxis methods find defects that TSP teams by themselves typically miss.
5. Since the Praxis methods employ special proprietary tools and require considerable skill, it would not be practical for the SEI to incorporate them into their offerings. However, the SEI and Praxis could work together to broaden the use of their methods. Based on the analyses done to date, it appears that combining elements of the TSP and the Praxis methods could produce significantly improved results.

3 Implications for the TSP

To broaden the TSP initiative to include safety-critical systems, the following tasks should be considered:

1. Produce a set of process scripts to define the STAMP method as currently used in accident investigations.
2. Obtain the services of Nancy Leveson or members of her group to review these scripts and ensure that they properly represent the STAMP method.
3. Based on the knowledge gained from Steps 1 and 2, define and produce the scripts needed for a TSP systems development process that incorporated STAMP principles.
4. Again obtain the services of Nancy Leveson or members of her group to review these scripts and ensure that they properly represent the STAMP method.
5. Based on the results of the preceding steps, use the STAMP method to develop a feedback-control model of the resulting TSP systems development process.
6. Based on the results of Step 5, design and develop the project reporting and management control systems for guiding and controlling design and development processes.
7. As appropriate, incorporate these process enhancements in the TSP process and test them with actual systems development projects.
8. Where these enhancements prove effective, incorporate them in the TSP process materials, the PSP and TSP course offerings, and the SEI certification criteria.
9. Establish and maintain a continuing relationship with Praxis High Integrity Systems, with the objective of facilitating the joint use of their methods with those developed by the SEI. While TSP would be an obvious candidate for this activity, Praxis methods could be usefully incorporated into other SEI technologies.

4 Conclusion

While the TSP process as presently embodied could be very useful for developing safety-critical systems, the investigation conducted during development of this report found that the current best practices for safety-critical incident investigations are broadly applicable to other processes. The two areas that appear to be of most immediate value are in process design and definition and in defining the management metrics needed for monitoring and controlling the systems development process.

At present, the Praxis design and development methods for high-integrity software represent a unique and proven capability. By working jointly, the SEI and Praxis could more closely relate the two organization's methodologies to accelerate their widespread adoption.

Bibliography

URLs are valid as of the publication date of this document.

[Benediktsson 01] Benediktsson, O.; Hunter, R. B.; & McGettrick, A. D. "Processes for Software in Safety Critical Systems." *Software Process Improvement and Practice 6*, 1 (2001): 47–62. http://www.hi.is/~oddur/publicat/01/61508Spice.pdf

[Goddard 00] Goddard, Peter L. "Software FMEA Techniques," 118–123. *2000 Proceedings Annual Reliability and Maintainability Symposium.* Los Angeles, CA, Jan. 24–27, 2000. New York, NY: IEEE Computer Society Press, 2000.

[Johnson 03] Johnson, C. W. & Holloway, C. M. *The ESA/NASA SOHO Mission Interruption: Using the STAMP Accident Analysis Technique for a Software Related 'Mishap.'* http://www.dcs.gla.ac.uk/~johnson/papers/STAMP.pdf (2003).

[Lees 96] Lees, B. & Jenkins, D. G. "Supporting Software Quality in an Integrated Safety-Critical Systems Development Environment." *Software Quality Journal 5*, (1996): 117–125.

[Leveson 95] Leveson, Nancy G. *Safeware: System Safety and Computers.* Reading, MA: Addison Wesley, 1995 (ISBN 0201119722). http://www.awprofessional.com/title/0201119722

[Leveson 02a] Leveson, Nancy G. "A Systems Model of Accidents," 476–486. *Proceedings of the 20th International System Safety Conference.* Edited by J.H. Wiggins & S. Thomason. Denver, CO, Aug. 5–9, 2002. Unionville, VA: International Systems Safety Society, 2002.

[Leveson 02b] Leveson, Nancy G.; Allen, P.; & Storey, M. A. "The Analysis of a Friendly Fire Accident using a Systems Model of Accidents." *Proceedings of the 20th International System Safety Conference.* Edited by J.H. Wiggins & S. Thomason. Denver, CO, Aug. 5–9, 2002. Unionville, VA: International Systems Safety Society, 2002. http://sunnyday.mit.edu/accidents/issc-bl-2.pdf

[Leveson 02c] Leveson, Nancy G. *A New Approach to System Safety Engineering.* http://sunnyday.mit.edu/book2.pdf (2002).

[Madan 97] Madan, Sanjay. "Techniques to Facilitate Development of Safety Critical Software Systems," 249–252. *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) '97.* St. John's, Newfoundland, Canada, May 25–28, 1997. IEEE, 1997.

[Maier 97] Maier, Thomas. "FMEA and FTA to Support Safe Design of Embedded Software in Safety-Critical Systems" 351–367. *Safety and Reliability of Software Based Systems (Twelfth Annual CSR Workshop).* Edited by Roger Shaw. Bruges, Belgium, Sept. 12–15, 1995. Springer–Verlag, 1997 (ISBN 3540760342).

[Stalhane 98] Stalhane, Tor & Wedde, Kari Juul. "Modification of Safety Critical Systems—An Evaluation of Three Approaches." *Microprocessors and Microsystems 21*, 10 (April 1998): 611–619.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE November 2005	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Safety-Critical Systems and the TSP		5. FUNDING NUMBERS FA8721-05-C-0003
6. AUTHOR(S) Watts S. Humphrey		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2005-TN-011
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES		
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

Because the Team Software Process (TSP) has proven effective for developing high-quality software applications, a brief review of the safety-critical systems field has been conducted to determine whether the TSP could be usefully extended into this area. This technical note provides a brief overview of recent work in software safety, discusses the problems and implications of using the TSP for developing safety-critical systems, and presents some conclusions. This information is relevant to software developers and acquirers of safety-critical software-intensive systems.

14. SUBJECT TERMS metric, quality, software development, software engineer, software-intensive system, TSP			15. NUMBER OF PAGES 18
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18 298-102